Steven M. Hoffberg

From: Steven M. Hoffberg [steve@hoffberg.org]

Sent: Thursday, November 18, 2004 12:18 PM

To: 'Nguyen, Nga'

Subject: 09/599,163 token.c

```
/* ----------------------------------------------------------------------
 * TVS Token handling
 * Copyright (c) 1995 Newshare Corporation
 * ----------------------------------------------------------------------
 */

#define TOKEN_IMPLEMENTATION                  /* leave this at the top */

#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include <errno.h>
#include <ctype.h>
#include <time.h>
#include <sys/types.h>
#include <netinet/in.h>                      /* to swap byte order */

#include "tvs_util.h"                    /* some tvs externals */
extern int tvs_sizeof_profile();

/* token "flags" data structure
 * (future use -- currently just a "magic" data structure)
 */

struct token_flags {
   unsigned int version: 4;
   unsigned int unused: 14;
   unsigned int never:  14;
};

typedef struct token_flags *TOKEN_FLAGS;

union tf_bytes {
   struct token_flags tfs;
   char tfc[sizeof(struct token_flags)];
};

/* token data structure */

struct user_token {
   unsigned short     token_version;
   unsigned short     server_id;
   unsigned int       profile[6];        /* DAMAGE CONTROL !!!!!!!!!!!!! */
   struct token_flags  t_flags;
   time_t             st_time;
};

typedef struct user_token *TVS_TOKEN;

union tk_bytes {
   struct user_token toks;
   char tokc[sizeof(struct user_token)];
};

/*
 * I have to include this AFTER my internal definitions, not before!!!
 */

#include "token.h"

#define PUBLIC
#define PRIVATE static

#define TVS_TOKEN_VERSION  1
#define TOKEN_FLAG_VERSION 1

#define MALLOC(x) malloc(x)

/* ----------------------------------------------------------------------
 * generate a new token data structure
 * ----------------------------------------------------------------------
 */

PRIVATE TVS_TOKEN
tvss_new_token()
{
  TVS_TOKEN new;

  new = (TVS_TOKEN) malloc(sizeof(struct user_token));
  if (!new) return (TVS_TOKEN) NULL;

  memset((char *)new, 0, sizeof(struct user_token));

  new->token_version = TVS_TOKEN_VERSION;
  return new;
}


/* ----------------------------------------------------------------------
 * create a new (unencoded) token-flags data structure
 * ----------------------------------------------------------------------
 */

#define TOKEN_MAGIC_NEVER  0x1555
#define TOKEN_MAGIC_UNUSED 0x2aaa

PUBLIC TOKEN_FLAGS
tvss_new_flags()
{
  TOKEN_FLAGS new;

  new = (TOKEN_FLAGS) malloc(sizeof(struct token_flags));
  if (!new) return (TOKEN_FLAGS) NULL;

  memset((char *)new, 0, sizeof(struct token_flags));

  new->version = TOKEN_FLAG_VERSION;
  new->never   = TOKEN_MAGIC_NEVER;
  new->unused  = TOKEN_MAGIC_UNUSED;

  return new;
}

/* ------------------------------------------------------------------------
 * create AND FILL a new (unencoded) token data structure
 * ------------------------------------------------------------------------
 */

PUBLIC TVS_TOKEN
tvss_make_token(unsigned short s_id, TVS_PROFILE prof, TOKEN_FLAGS flags)
{
  TVS_TOKEN new;

  new = tvss_new_token();

  tvss_set_profile(new, prof);
  tvss_set_server_id(new, s_id);
  tvss_set_flags(new, flags);
  tvss_set_sttime(new);

  return new;
}

/* ------------------------------------------------------------------------
 * insert the user profile
 * ------------------------------------------------------------------------
 */

PUBLIC void
tvss_set_profile(TVS_TOKEN token, TVS_PROFILE prof)
{
  memcpy(token->profile, prof, tvs_sizeof_profile());
}

/* -----------------------------------------------------------------------
 * retrieve the token's user profile
 * -----------------------------------------------------------------------
 */

PUBLIC int
tvss_get_profile(TVS_TOKEN token, TVS_PROFILE prof)
{
  memcpy(prof,  token->profile, tvs_sizeof_profile());
  return 1;
}

/* -----------------------------------------------------------------------
 * set the token's server id
 * -----------------------------------------------------------------------
 */

PUBLIC void
tvss_set_server_id(TVS_TOKEN token, unsigned short sid)
{
  token->server_id = sid;
}

/* -----------------------------------------------------------------------
 * get the token's server id
 * -----------------------------------------------------------------------
 */

PUBLIC unsigned short
tvss_get_server_id(TVS_TOKEN token)
{
 return token->server_id;
}


/* -----------------------------------------------------------------------
 * set the token's start time
 * -----------------------------------------------------------------------
 */

PUBLIC void
tvss_set_sttime(TVS_TOKEN token)
{
  token->st_time = htonl((long) time((time_t) NULL));
}

/* -----------------------------------------------------------------------
 * get the token's start time
 * -----------------------------------------------------------------------
 */

PUBLIC time_t
tvss_get_sttime(TVS_TOKEN token)
{
 return ntohl(token->st_time);
}

/* ----------------------------------------------------------------------
 * insert the token flags
 * ----------------------------------------------------------------------
 */

PUBLIC void
tvss_set_flags(TVS_TOKEN token, TOKEN_FLAGS flags)
{
  memcpy((char *)&(token->t_flags), flags, sizeof(struct token_flags));
}

/* ----------------------------------------------------------------------
 * retrieve the token's user profile
 * ----------------------------------------------------------------------
 */

PUBLIC int
tvss_get_flags(TVS_TOKEN token, TOKEN_FLAGS flags)
{
  memcpy(flags, (char *)&(token->t_flags), sizeof(struct token_flags));
  return 1;
}

/* ----------------------------------------------------------------------
 * tvss_token_fmt_is_valid - make sure we dont receive just barf!
 * ----------------------------------------------------------------------
 */

PUBLIC int
tvss_token_fmt_is_valid(TVS_TOKEN token)
{
  if (token->token_version != TVS_TOKEN_VERSION) return 0;

  if (token->t_flags.version != TOKEN_FLAG_VERSION) return 0;
  if (token->t_flags.unused  != TOKEN_MAGIC_UNUSED) return 0;
  if (token->t_flags.never   != TOKEN_MAGIC_NEVER)  return 0;

  return 1;
}

/* ----------------------------------------------------------------------
 * encode it
 * ----------------------------------------------------------------------
 */

#define MAX_TOKEN_LENGTH 128

PUBLIC char *
tvss_encode_token(TVS_TOKEN token)
{
  char *string;
  int len = MAX_TOKEN_LENGTH;

  string = (char *) MALLOC(len);
  if (!string) return (char *) NULL;

  if (!tvs_encode((char *) token, sizeof(struct user_token), string, &len))
   return (char *) NULL;

  string[len] = '\0';
  return (char *)string;
}

/* -------------------------------------------------------------------------
 * decode it
 * -------------------------------------------------------------------------
 */

PUBLIC TVS_TOKEN
tvss_decode_token(char *string)
{
  int len;
  TVS_TOKEN token;

  len = sizeof(struct user_token);
  token = tvss_new_token();

  if (!tvs_decode(string, (char *) token, &len))
   return (TVS_TOKEN) NULL;

  return token;
}
```

Very truly yours,

Steven M. Hoffberg
Milde & Hoffberg, LLP
Suite 460
10 Bank Street
White Plains, NY 10606
(914) 949-3100 tel.
(914) 949-3416 fax
steve@hoffberg.org
www.hoffberg.org

Confidentiality Notice: This message, and any attachments thereto, may contain confidential information which is legally privileged. The information is intended only for the use of the intended recipient, generally the individual or entity named above. If you believe you are not the intended recipient, or in the event that this document is received in error, or misdirected, you are requested to immediately inform the sender by reply e-mail at Steve@Hoffberg.org and destroy all copies of the e-mail file and attachments. You are hereby notified that any disclosure, copying, distribution or use of any information contained in this transmission other than by the intended recipient is strictly prohibited.